Interim Report on Contract NAS8-35354

SPACE RADIATION STUDIES

(NASA-CR-171410) SPACE RADIATION STUDIES Interim Report, Jul. 1984 - Jan. 1985 (Alabama Univ., Huntsville.) 13 p
HC A02/MF A01 CSCI 03B G3/93

N85-26564
Unclas
15188

For the Period July 1984 - January 1985

Prepared By

Cosmic Ray Laboratory

The University of Alabama in Huntsville
Huntsville, Alabama 35899

5 March 1985

SUMMARY

During this period an intensive effort has been made to develop the data analysis software for the Nuclear Radiation Monitor to fly on Spacelab - 2. Results of data analysis of the Active Radiation Detector which flew on Spacelab - 1 will be presented later this year.

NRM Data Analysis Software

Development of NRM data analysis software is in progress. The overall data flow diagram (Figure 1) has been revised in the light of better understanding of the format in which the data will be delivered to MSFC and the capabilities of the hardware environment in which the analysis programs will operate. The use of structured techniques for the software design appears to be working well. Figure 2 shows an example of one of the processes in Figure 1 after it has been broken down into its component processes. The present technique uses PASCAL as a Program Development Language (pseudo-code) to develop and document the software design. Attachment A is an example of the PASCAL pseudo-code written to implement the processes in Figure 2. The pseudo-code is sufficiently general that the actual code may be written from this design in almost any programming language. Note also that the pseudo-code may evolve in its details as the overall system design matures. UAH Research Professor W. Paciesas is supervising this software development. We intend to have portions of the software ready for use with tapes of sample data which are scheduled to be delivered to MSFC by early May 1985. The full system is scheduled for completion in time for the expected delivery of the Spacelab 2 mission data tapes (early September).

NRM DATA ANALYSIS SYSTEM
DATA FLOW

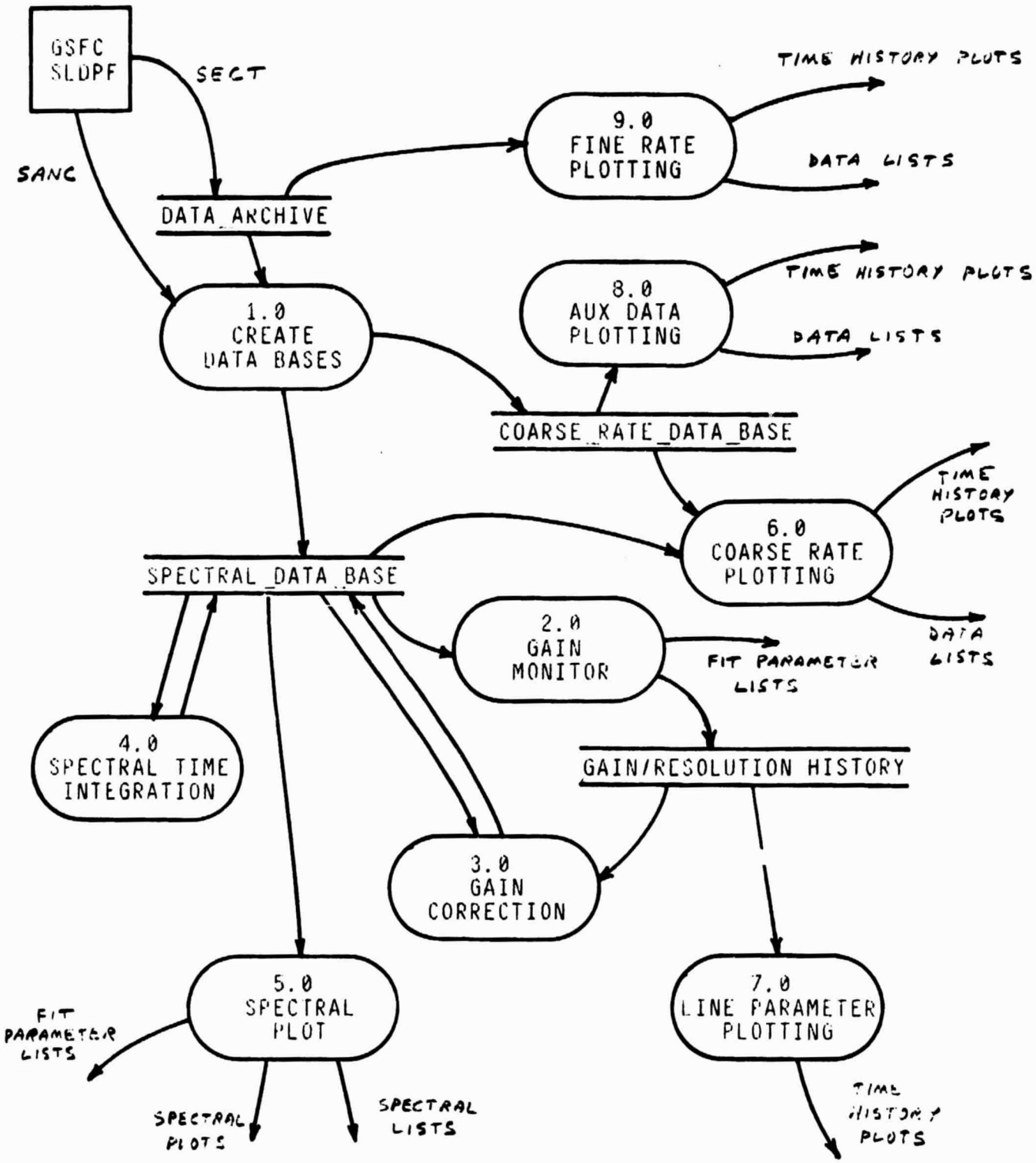

Figure 1

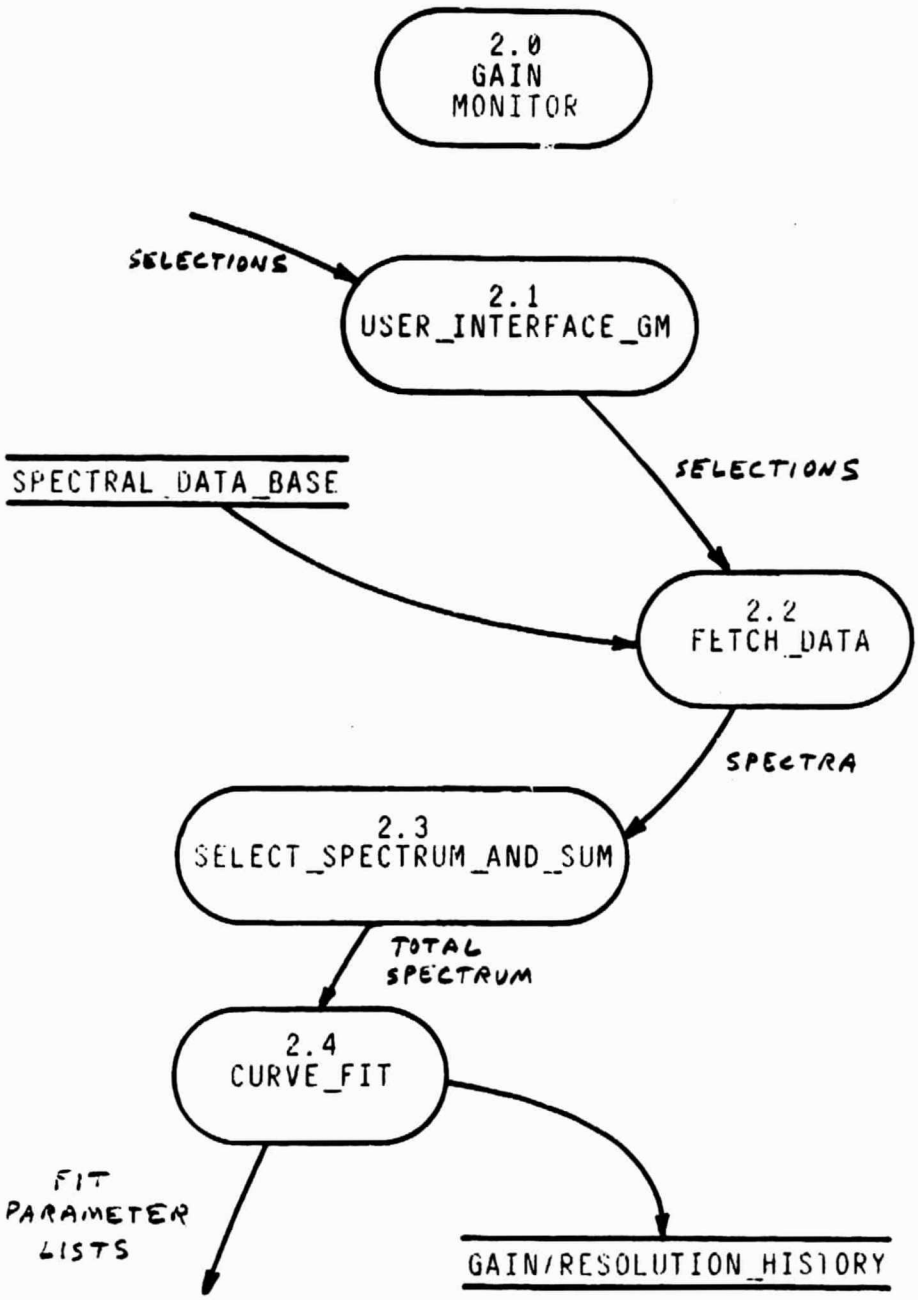
2.0
GAIN
MONITOR
SELECTIONS
2.1
USER_INTERFACE_GM
SELECTIONS
SPECTRAL_DATA_BASE
2.2
FETCH_DATA
SPECTRA
2.3
SELECT_SPECTRUM_AND_SUM
TOTAL
SPECTRUM
2.4
CURVE_FIT
FIT
PARAMETER
LISTS
GAIN/RESOLUTION_HISTORY

Figure 2

ATTACHMENT A

Updated: 2/11/83

```
(*************************************************************************)
(***************** GAIN MONITOR PROCEDURES ***********************)
(*************************************************************************)
```

{ The following are preliminary blocked-out procedures intended to implement PROCESS 2. in the NRM data flow diagram.}

```
(*************************************************************************)
(***************** GAIN MONITOR PROCEDURES ***********************)
(*************************************************************************)
```

```
(*************************************************************************)
PROCEDURE   GAIN_MONITOR;                    {main procedure}
(*************************************************************************)

{PROCESS NUMBER - 2}

{DESCRIPTION:   This procedure compiles a history of NRM gain para-
                meters and resolution as a function of time during
                the mission.}


TYPE    CHANNELS_C = ARRAY [0..511] OF INTEGER;
        CHANNELS_R = ARRAY [0..511] OF REAL;
        CHAR_3 = PACKED ARRAY [1..3] OF CHAR;
        CHAR_4 = PACKED ARRAY [1..4] OF CHAR;


{The following variable declarations may become global to the entire
 main program at a later stage of program design}

{This procedure, GAIN_MONITOR, assumes all global variables used in
 internal procedures to act as PASSED VAR PARAMETERS - (or as declared
 as COMMON variables in FORTRAN).  This is due to the large number of
 variables to pass as VAR parameters and to the current ambiguity of
 data types and implementation language to be used.}

VAR

    RO_NUM : INTEGER;               {# of readouts per integration period}
    START_TIME : REAL;              {mission time to begin program run}
    STOP_TIME : REAL;               {mission time to stop program run}
    LO_CHANNEL : INTEGER;           {lower channel number of interest}
    HI_CHANNEL : INTEGER;           {higher channel number of interest}
    SPECTRUM_TYPE : CHAR_3;         {'TOT','PAC','PC'}
    GAIN_RANGE : CHAR_3;            {'LO','MED','HI'}
    FUNCTION_TYPE_G : CHAR_4;       {'EXP','POW','POLY'}
    CENTROID_G : REAL;              {initial guess of CENTROID parameter}
    WIDTH_G : REAL;                 {initial guess of WIDTH parameter}
    INTENSITY_G : REAL;             {initial guess of INTENSITY parameter}
    IP_RATE : CHANNELS_R;           {stores rate for each channel}
    IP_ERROR : CHANNELS_R;          {stores error in calculation per channel}
    IP_START : REAL;                {time each integration period starts}
    IP_STOP : REAL;                 {time each integration period stops}
    LIVE_TIME : REAL;               {holds current readout's live time}
    UP_TIME : REAL;                 {sum of live times for integ. period}
    CURR_TIME : REAL;               {mission time current readout begins}
    PC_COUNT : CHANNELS_C;          {stores PC counts for each channel over
                                                integration period}
    PAC_COUNT : CHANNELS_C;         {stores PAC counts for each channel over
                                                integration period}
```

```
{=============================================================)
PROCEDURE  USER_INTERFACE_GM  (VAR  {TBD} );
{=============================================================)

{PROCESS NUMBER - 2.1}

{DESCRIPTION:  This process allows user to change default values,
               or prompts the user for a set of user-selections.
               These selections will be passed as parameters to
               later routines.  }

BEGIN

     {set default values for appropriate user-selected variables}
     {display menu informing user of his options.}
     {check user selections for validity}

     WHILE {any user selections are invalid} DO
     BEGIN
        {prompt user for re-entry of invalid selections};
        {check user selections for validity}
     END;

     {store user's options in appropriate parameters }

END;   {procedure USER_SELECTIONS_GM}

{=============================================================)
```

```
{=============================================================}
PROCEDURE   FETCH_DATA (VAR CURR_TIME,UPTIME : REAL;
                        VAR PC_COUNT,PAC_COUNT : CHANNELS;
                        GAIN_RANGE : CHAR_3);
{=============================================================}

{PROCESS NUMBER - 2.2}

{DESCRIPTION: This procedure retrieves data from SPECTRAL_DATA_BASE
              (i.e. secondary storage) to make it available for
               program use.}

VAR
   I : INTEGER;
   COUNT : INTEGER;

BEGIN

     {data other than that accessed below may be added later.}

     {the following code assumes sequential data within each GAIN_RANGE
          of PAC and PC.}
     {the following code also assumes that mission time from GMT and
          live time are present in SPECTRAL_DATA_BASE.}


     READ (CURR_TIME);
     READ (UPTIME);

     {move to GAIN_RANGE in spectral data portion of 20.16s readout}

     FOR I := 0 TO 511 DO
     BEGIN
          READ(COUNT);
          PC_COUNT[I] := COUNT;
     END; {for loop}

     FOR I := 0 TO 511 DO
     BEGIN
          READ(COUNT);
          PAC_COUNT[I] := COUNT;
     END; {for loop}

END;  {procedure FETCH_DATA}

{=============================================================}
```

```
{=============================================================}
PROCEDURE  SELECT_SPECTRUM_AND_SUM  (VAR {TBD}   );
{=============================================================}

{PROCESS NUMBER 2.3}

{DESCRIPTION:  }

VAR

     I : INTEGER;                    {loop control variable}
     TOTAL_COUNT : CHANNELS;         {total event count for each channel of
                                          interest over integration period}


BEGIN

     WHILE  CURR_TIME < IP_STOP  DO
     BEGIN
          FETCH_DATA (CURR_TIME,UPTIME,PC_COUNT,PAC_COUNT,GAIN_RANGE);
          UP_TIME := UP_TIME + LIVE_TIME;

          FOR I := LO_CHANNEL  TO  HI_CHANNEL  DO

              IF SPECTRUM_TYPE = 'TOT' THEN
                 TOTAL_COUNT[I] := TOTAL_COUNT[I] + PC_COUNT[I]
                                                  + PAC_COUNT[I]
              ELSE
                 IF SPECTRUM_TYPE = 'PAC' THEN
                    TOTAL_COUNT[I] := TOTAL_COUNT[I] + PAC_COUNT[I]
                 ELSE
                    (* spectrum_type is 'PC' *)
                    TOTAL_COUNT[I] := TOTAL_COUNT[I] + PC_COUNT[I];

     END;  {while loop}

     FOR I := LO_CHANNEL  TO  HI_CHANNEL  DO
     BEGIN
          IP_RATE[I] := TOTAL_COUNT[I] / LIVE_TIME;
          IP_ERROR[I] := SQRT(IP_RATE[I]);
     END;  {for loop}

END;  {procedure  S_S_A_S}

{=============================================================}
```

```
{=================================================================}
PROCEDURE  CURVE_FIT  (VAR  {TBD}  )
{=================================================================}

{PROCESS NUMBER - 2.4}

{DESCRIPTION:  ...}

BEGIN
     {algorithm TBD}
     {will need to fit results to a given function based on initial
      parameter guesses}
     {if error is within bounds then
                         update all guesses
        else
            set error flag}

END;  {procedure CURVE_FIT}

{=================================================================}
```

```
(*************************************************************************)
BEGIN            {main procedure GAIN-MON}
(*************************************************************************)

USER_INTERFACE_GM;

{open GAIN_RESOLUTION_HISTORY file}

(* assumming SPECTRAL_DATA_BASE is sequentialfile,following loop is
    used to move to desired data *)

REPEAT
     FETCH_DATA (CURR_TIME, UPTIME, PC_COUNT, PAC_COUNT,GAIN_RANGE);
UNTIL  CURR_TIME >= START_TIME;

IP_START := CURR_TIME;
IP_STOP := IP_START + (20.16 * RO_NUM);
LIVE_TIME := 0;

WHILE  STOP_TIME > IP_STOP  DO
BEGIN
     SELECT_SPECTRUM_AND_SUM ({TBD});
     CURVE_FIT ({TBD});

     {store results from CURVE_FIT into GAIN_RESOLUTION_HISTORY
           including  appropriate user selected variables}


     IP_START := IP_STOP;
     IP_STOP := IP_START + (20.16 * RO_NUM);
     LIVE_TIME := 0;

END;  {while loop}

{close GAIN_RESOLUTION_HISTORY file}

(*************************************************************************)
END;  {main procedure GAIN_MONITOR}
(*************************************************************************)
```